REMARKS

This amendment is responsive to the Office Action dated October 23, 2006. Applicant has amended claims 1, 13, 14, 26 and 27. Claims 1-30 remain pending.

Claim Rejection Under 35 U.S.C. § 112

In the Office Action, the Examiner rejected claims 13 and 26 under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Applicant has amended claims 13 and 26 for purposes of clarification. Applicant submits that claims 13 and 26, as amended, particularly point out and distinctly claim the subject matter, as required by 35 U.S.C. 112, second paragraph.

Claim Rejection Under 35 U.S.C. § 102

In the Office Action, the Examiner rejected claims 1-30 under 35 U.S.C. 102(e) as being anticipated by Talwar et al. (US 2004/0230949 A1). Applicant respectfully traverses the rejection. Talwar fails to disclose each and every feature of the claimed invention, as required by 35 U.S.C. 102(e), and provides no teaching that would have suggested the desirability of modification to include such features.

For example, with respect to claim 1, Talwar fails to teach or suggest a method comprising processing read instructions with an emulated processor executing within an emulation environment to output independent read requests via an operand interface and an op-code interface of the emulated processor. Talwar also fails to teach or suggest outputting the independent read requests via the operand interface and the op-code interface of the emulated processor to independently fetch op-codes and operands from an emulated memory external from the emulated processor, as required by claim 1. Talwar also fails to teach or suggest verifying that the emulated memory properly executed the independent read requests by independently comparing op-code reference data and operand reference data to the operands and the op-codes received in response to the read requests, as required by claim 1. These arguments are also substantially applicable to independent claims 14 and 27.

Talwar describes an emulation verification system that includes a verifier to determine the validity of native language code. FIG. 1A and [0027], lines 4-8. That is, the Talwar system prevents illegitimate or malicious tampering with native language information associated with emulation applications (e.g., Java applications). Paragraph [0024]. In some instances, the verifier may assess the validity of multiple native language codes residing in separate native language shared libraries. FIG. 1C and [0035], lines 3-8. Native language code may include native language instructions, such as a C language function and/or subroutine. [0031], lines 3-6. The native language shared libraries may also include bytecode verification information, which comprises an encoded value corresponding to bytecodes stored within emulation language class information. [0032], lines 10-12.

To verify the native language code, the emulation verification system loads emulation language class information, which may comprise bytecodes that provide instructions and data for execution and processing by the emulation verification system. FIG. 2A and [0037], lines 13-17. Each of these bytecodes may include both an op-code that indicates the operation to be performed and an operand that provides information associated with the operation. [0042], lines 3-6. In instances where multiple native language codes are verified, bytecodes corresponding to each of the separate native language codes may be loaded. FIG. 1C and [0035]. Upon loading the bytecodes, emulation verification system proceeds to load and verify the native language code according to a native language verification process. [0039], lines 6-9.

The Examiner supports the rejection of Applicant's claim 1 by reference to [0042], stating that the emulation verification system of Talwar executes bytecode instructions included in the emulation class information, that each bytecode may include an op-code and an operand, and that the emulation verification system fetches the bytecode and performs the instruction indicated by the op-code included within the bytecode. The Examiner further supports the rejection of Applicant's claim 1 by reference to elements 132A and 132B of FIG. 1C, stating that "different interfaces" exist to fetch bytecodes from "two different shared libraries." Based on this reasoning, the Examiner appears to conclude that the Talwar emulation verification system independently fetches op-codes and operands from the shared libraries.

Applicant submits that the Talwar emulation verification system is entirely different from the Applicant's claims. Applicant's claims are directed to emulation of a particular type of

microprocessor, i.e., a microprocessor in which an instruction processor has separate interfaces to fetch op-codes and operands. That is, the microprocessor has separate interfaces by which op-codes and operands can be independently fetched. The claims are directed to embodiments for emulating such processors to test and verify the separate interfaces and the memory structures to which they connect.

These features are specifically recited in the claims. For example, as discussed above, claim 1 requires a method comprising processing read instructions with an emulated processor executing within an emulation environment to output independent read requests via an operand interface and an op-code interface of the emulated processor. That is, claim 1 literally requires outputting independent read requests (i.e., separate requests) over two different interfaces (i.e., an operand interface and an op-code interface) of an emulated processor. Thus, the emulated processor of claim 1 must have these two distinct interfaces, and the method requires outputting independent requests over the interfaces.

Claim 1 also requires outputting the independent read requests via the operand interface and the op-code interface of the emulated processor to independently fetch op-codes and operands from an emulated memory external from the emulated processor. Thus, claim 1 requires that the op-codes and operands are independently fetched, and that they are fetched from an emulated memory. Further, claim 1 requires verifying that the emulated memory properly executed the independent read requests by independently comparing op-code reference data and operand reference data to the operands and the op-codes received in response to the read requests.

With respect to these elements, the Examiner states execution engine 234 of Talwar is an emulated processor that executes bytecode instructions. The Examiner states that each bytecode "includes an op-code ... and can include an operand. Execution engine 234 fetches an op-code and [operand and/or data]." The Examiner then refers to elements of FIG. 1C 132A and 132B as "different interfaces" to two different shared libraries and concludes that these teachings anticipate Applicant's claims. The Examiner's reasoning with respect to Talwar is flawed for several reasons.

First, the fact that the execution engine 234 of Talwar accesses different native shared libraries to retrieve different bytecode instructions does not teach or suggest that operands and

op-codes for those bytecodes are retrieved using different interfaces. By the Examiner's own admission, each individual bytecode in Talwar contains an op-code and any corresponding operands. Thus, retrieving one bytecode from one native library and a different bytecode from a second native library would not suggest that the operands for the bytecodes are somehow fetched using a different memory interface than the opcodes for those same bytecodes. To the contrary, the bytecode from the first library would contain both its opcode and any operand and any "interface" to the first library retireves that bytecode, i.e., both the opcode and the operand. Similarly, any bytecode read from the second library contain both its opcode and any operand, and any "interface" to the second library to retrieve the entire bytecode would retrieve both the opcode and the operand.

Second, the Examiner's reliance on FIG 1C as teaching use of separate interfaces at all is misplaced. FIG. 1C of Talwar appears to represent a logical diagram that is not representative of the interface between an emulated processor and an external memory architecture, as recited in Applicant's claim. Specifically, Applicant's claim 1 requires an emulated processor having an operand interface and an op-code interface to independently fetch op-codes and operands from an emulated memory external from the emulated processor.

Contrary to the Examiner's position, paragraph [0045] and FIG. 3A makes clear that the Talwar computer system uses a ***single*** communication bus 357 that connects the emulated processor (operating on processor 351) to the memory that would store the byte-codes of the native shared libraries (stored in memory 352 or bulk storage 354). Thus, to the extent Talwar's execution engine 234 can even be construed as an emulated processor, only communications bus 357 couples it to any external memory. Talwar clearly does not describe an emulated processor having an operand interface and an op-code interface to independently fetch op-codes and operands from an emulated memory external from the emulated processor. To the contrary, Talwar clearly suggests that the emulation verification system fetches a single encoded bytecode over a single interface. Thus, Talwar does not teach or suggest that separate emulated interfaces exist by which the emulated processor may independently fetch op-codes and operands, as required by Applicant's claim 1.

Third, Applicant's claim 1 requires an *emulated memory* external to the emulated processor. Applicant's can find no feature in Talwar that constitutes an emulated memory.

Memory 352 and bulk storage 354 are described as real storage devices, and neither the Java bytecode nor the native language instructions that may be stored in the devices emulate a memory in any manner. For this reason, the Talwar emulation verification system for verifying native language code is entirely unrelated to a system for emulating an instruction processor and the processor's operand and op-code interfaces to test an emulated memory architecture external to the emulated instruction processor, as recited in Applicant's claims.

For at least these reasons, Talwar fails to establish a prima facie case for anticipation of Applicant's claims 1-30 under 35 U.S.C. 102(e). Applicant respectfully requests withdrawal of this rejection.

CONCLUSION

All claims in this application are in condition for allowance. Applicant respectfully requests reconsideration and prompt allowance of all pending claims. Please charge any additional fees or credit any overpayment to deposit account number 50-1778. The Examiner is invited to telephone the below-signed attorney to discuss this application.

Date: January 19, 2007

By: Kent Sieffert

SHUMAKER & SIEFFERT, P.A.
8425 Seasons Parkway, Suite 105
St. Paul, Minnesota 55125
Telephone: 651.735.1100
Facsimile: 651.735.1102

Name: Kent J. Sieffert
Reg. No.: 41,312